Claims

1. An interface for interfacing between front-end data processing systems and back-end data processing systems, the interface comprising an engine, a node layer comprising at least one node, and a utility layer comprising at least one utility, and in which:

the engine comprises means for receiving a message containing a request from a front-end system for a transaction to be performed by a back-end system, and means for interpreting said message to select a relevant node for interfacing,

each node represents business logic interfaces to a back-end system,

each node exposes business logic capabilities to the engine;

the engine comprises means for using the exposed node business logic capabilities to build a process map linking received request messages with nodes;

each utility is coupled as a proxy to a back-end system, comprises means for receiving a transaction request from a node, for converting said request to a back-end system request, for receiving a response from the back-end system, and for routing a response to the requesting node,

each node comprises means for routing a received response to the engine; and

the engine comprises means for routing a response to the requesting front-end system.

2. An interface as claimed in claim 1, wherein the engine comprises means for dynamically maintaining the process map according to the exposed node business logic capabilities.

3. An interface as claimed in claim 2, wherein the process map comprises a script file.

4. An interface as claimed in claim 3, wherein the process map comprises script messages, each message having a map associating incoming parameter names with standardised names.

5. An interface as claimed in claim 4, wherein each message of the process map specifies an associated node, a list of the parameters the node requires, and values which it returns for a type of incoming message.

6. An interface as claimed in claim 1, wherein the utilities comprise means for interfacing with the node layer according to a uniform interface model.

7. An interface as claimed in claim 1, wherein the engine comprises means for calling a plurality of nodes for a transaction request.

8. An interface as claimed in claim 7, wherein the engine comprises means for calling nodes in sequence, and for passing the output from a previous node to a next node.

9. An interface as claimed in claim 1, wherein the engine and each node comprise means for using a hashtable mapping keys to values for passing data and control to each other.

10. An interface as claimed in claim 9, wherein the engine and the nodes each comprise means for using a hashtable for returning a result from a back-end system.

11. An interface as claimed in claim 10, wherein the engine comprises means for requesting a return value for a transaction, and each node comprises means for defaulting to not passing a return value if one is not so requested.

12. An interface as claimed in claim 1, wherein each of the engine and each node comprise an object instantiated from an object-oriented class.

13. An interface as claimed in claim 12, wherein each of the engine and each node comprises means for using a hashtable which maps keys to values for passing data and control to each other, and the engine comprises means for passing a hashtable as a parameter in an execute method, a commit method, and a rollback method of a node object.

14. An interface as claimed in claim 12, wherein the engine comprises means for activating a sequence of nodes for a transaction, and each node comprises means for performing a rollback if a transaction fails.

15. An interface as claimed in claim 12, wherein the engine comprises an externally visible engine class, an object of which comprises means for instantiating:

a processor object for instantiating said node objects; and

a loader object for loading the process map, and for determining node objects associated with a received message.

16. An interface as claimed in claim 15, wherein the engine comprises means for instantiating a parser object for parsing a received message, for placing extracted data in a hashtable, and for returning the hashtable to the engine object.

17. An interface as claimed in claim 15, wherein the engine comprises a builder object comprising means for automatically updating the process map according to capabilities exposed by node classes.

18. An interface as claimed in claim 12, wherein each node class comprises a method for returning a string to the engine indicating the node capabilities.

19. An interface for interfacing between front-end data processing systems and back-end data processing systems, the interface comprising an engine, a node layer comprising at least one node, and a utility layer comprising at least one utility, and in which:

the engine comprises means for receiving a message containing a request from a front-end system for a transaction to be performed by a back-end system, and means for interpreting said message to select a relevant node for interfacing,

each node represents business logic interfaces to a back-end system,

each node exposes business logic capabilities to the engine;

the engine comprises means for using the exposed node business logic capabilities to build a process map linking received request messages with nodes;

each utility is coupled as a proxy to a back-end system, comprises means for receiving a transaction request from a node, for converting said request to a back-end system request, for receiving a response from the back-end system, and for routing a response to the requesting node,

each node comprises means for routing a received response to the engine,

the engine comprises means for routing a response to the requesting front-end system,

each of the engine and each node comprises an object instantiated from an object-oriented class, and

each of the engine and each node comprises means for using a hashtable which maps keys to values for passing data and control to each other, and the engine comprises means for passing a hashtable as a parameter in an execute method, a commit method, and a rollback method of a node object.

20. A method of interfacing between front-end data processing systems and back-end data processing systems, the method being performed by an interface comprising an engine for communicating with the front-end systems and a utility layer for communicating with the back-end systems, the method comprising the steps of:

the engine receiving from a front-end system a message incorporating a request for a transaction to be performed by a back end system but not indicating a particular back-end system suitable for the transaction,

the engine using a process map to select one of a plurality of nodes in a node layer which may provide a suitable link to the back-end systems for the request, the process map linking message types to nodes according to exposed business logic capabilities of the nodes,

the engine passing a request to the selected node,

the selected node communicating with a utility with which it is associated to instruct the utility to perform the transaction, receiving a response from the utility, and passing the response back to the node,

the node passing the response back to the engine, and the engine passing the response back to the requesting front-end.

21. A method as claimed in claim 20, wherein:

the engine dynamically creates a node object according to parameters retrieved from the process map,

the engine passes data from the received message to the created node, and

data is passed between the node object and the engine by passing a hashtable linking keys with associated data.

22. A method as claimed in claim 21, wherein the engine passes a hashtable as a parameter in an execute method, a commit method, and a rollback method, and the node rolls back according to the rollback method if the transaction fails.

23. A method as claimed in claim 20, wherein the process map is an XML script file.

24. A computer program product comprising software code for performing the steps of claim 20 when running on a digital computer